

REMARKS*Status of the Application*

Upon entry of this amendment, the Specification will have been amended to correct typographical errors in the originally filed application. The changes to the Specification are ***merely ministerial and do not add new matter*** and Claims 1-24 are pending in this case. Claims 1 and 2 stand rejected under 35 U.S.C. §103 (a) as allegedly being obvious over U.S. Patent 6,243,856 B1 (*Meyer et al.*) in view of U.S. Patent 6,061,743 (*Thatcher et al.*). Claims 3 and 4 stand rejected under 35 U.S.C. §103(a) as allegedly being obvious over *Meyer et al.* in view of *Thatcher et al.* and further in view of U.S. Patent 6,078,924 (*Ainsbury et al.*). Claims 5 and 6, 15-16, and 23 stand rejected under 35 U.S.C. §103(a) as allegedly being obvious over *Meyer et al.* in view of *Ainsbury et al.* Claims 7 and 8 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over *Meyer et al.* in view of *Ainsbury et al.* further in view of *Thatcher et al.* Claim 9 stands rejected under 35 U.S.C. §103(a) as being allegedly obvious over *Meyer et al.* in view of *Ainsbury et al.* and further in view of U.S. Patent 6,237,135 (*Timbol*). Claims 10-13 stand rejected under 35 U.S.C. §103(a) as being allegedly obvious over *Meyer et al.* in view of U.S. Patent 4,484,267 (*Fletcher*). Claims 17-18, and 20-22 stand rejected under 35 U.S.C. 103(a) as being allegedly obvious over *Ainsbury et al.* in view of *Fletcher*. Claim 19 stands rejected under 35 U.S.C. 103(a) as being allegedly obvious over *Ainsbury et al.* in view of *Fletcher*, further in view of Thatcher. Claim 24 stands rejected under 35 U.S.C. §103(a) in view of *Meyer et al.* in view of *Ainsbury et al.*, and further in view of U.S. Patent 6,199,082 (*Ferrel et al.*).

In view of the foregoing amendments and following remarks, Applicant respectfully requests reconsideration of the present application and an early Notice of Allowance.

35 U.S.C. § 103(a) Rejections

Prima Facie Obviousness

To establish a *prima facie* case of obviousness, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to modify the reference or to combine reference teachings. Further, there must be a reasonable expectation of success after combining the references the intended purpose of the invention is realized. **Lastly, the prior art reference (or references when combined) must teach or suggest all the claim limitations.** The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991). Applicant respectfully submits that a *prima facie* case of obviousness has not been made for claims 1-24 of the present application.

The Present Invention

The present invention contemplates a system and methods for optimizing the caching of classes. In one aspect of the invention, **only *selected*** elements in a class path are cached. These elements are ***selected*** as those that have associated mechanisms that provide notification of changes. In this manner, the integrity of the cache can be maintained with little overhead

involved in the independent tracking of the class path elements. By keeping separate caches, reconstruction of a cache due to a detected change is limited to that cache only. Specifically, the present invention, as claimed, *inter alia* is operable to locate classes in a class path by ***generating a cache*** of information relating to the classes in the class path, ***requesting a search*** of the class path, and ***searching the cache*** to satisfy the search request. Also, the present invention is capable of, *inter alia*, locating classes in a multi-element class path by generating a search request for desired requests within the multi-element class path, and ***independently*** satisfying the request in association with each element in the class ***such that at least one of the elements has a cache of information sufficient to satisfy the request for that element***. Further, the present invention contemplates, *inter alia*, creating caches for selected elements of a class path. In this context, the present invention parses the class path into names of elements, determines that elements are viable for caching, and creates caches for the viable elements. These contemplated functions are operable by a system as offered by the present invention having a class path manager that receives request for identification or enumeration of classes in a class path, a cache to cache a viable element of the class path, and a wrapper that receives requests (e.g. identification or enumeration of classes in the class path) from the class path manager and provides a transparent level of indirection to services that are specific to the cache viable element.

Differences from Meyer et al.

Meyer et al. disclose a system and method for efficiently coding an animation sequence

and converting a series of opcodes and associated opcodes into an array of integers. (See Meyer et al. ABSTRACT, Lines 1-3). In a given embodiment, the *Meyer* invention is directed toward the efficient handling of animated scenes in an interactive manner at a user's computer at a remote location. (Col. 2, Lines 26-31). The animated scene or sequence is provided in a highly extensible animation language (e.g. Java using Java classes) such that the sequence can easily be run on any of a variety of computers. (Col. 2, Lines 38-41).

In contrast to the inventions of Applicant's claims 1-24, however, the systems and methods taught by *Meyer et al.* **do not teach a system or methods that contemplate the creation of cache having information relating to the classes of a class path, requesting a search on the class path, and searching the created cache to satisfy the search request.** (See Claims 1-24 of the present application). Rather, *Meyer et al.* teach a system and method to efficiently deliver animated scenes, wherein the animated scene is compiled to make use of an efficient set of opcodes for rendering. The animation sequence is created through some authoring tool and then translated into a language (e.g. a scene definition language) used to describe the animation sequence. The scene definition language is then compiled to produce opcodes that are converted into an extensible markup language, such as Java including class files. (Col. 7, Lines 58-67; Col. 8, Lines 1-4). *Meyer et al.* goes further to teach that the package name of scene definition language file is defined by its location in the class path. The package name is used to create the full path to the file relative to the class path. (Col. 11, Lines 50-52). *Meyer et al. simply do not teach or even suggest the creation of a cache contained therein information*

relating to the class of a class path, the requesting and execution of search on the cache to locate a particular class.

As Meyer et al. does not teach every limitation of the present invention, a *prima facie* case of obviousness has not been made. Moreover, Applicant submits that the present invention is not obvious over the *Meyer et al.* reference when combined with the other herein described cited references since there is no motivation in the *Meyer et al.* reference to combine with the other herein described references. *Meyer et al.* is directed to a system and methods for efficiently coding an animation sequence. The present invention, as stated, is directed to optimizing class identification for a class path. One of ordinary skill in the art would find it difficult to, refer to, yet alone, combine the teachings found in *Meyer et al.* with the other herein described references to teach the present invention. Further, even if the *Meyer et al.* reference is combined with the other herein described references, the present invention is not obvious over these herein described references since these herein described references, alone, or in combination **fail to teach every limitation of the present invention**. For these reasons, Applicant respectfully requests that the obviousness rejection be withdrawn.

Differences from *Thatcher et al.*

Thatcher et al. disclose a computer system having a first namespace with a target, and an interface module operative to read the first namespace. A namespace as described in the *Thatcher et al.* reference is a collection of names in a computing environment. (Col. 1 Line 24).

Sometimes, each of the various names in a namespace are referred to as objects. A second interface module is operative to read the second namespace. The system maintains a registry that comprises information that associates the first name space with the first interface module, the second namespace with the second interface module, and the target with the second interface module. A generic user interface may be used to access the registry and determine the associations therein. (See *Thatcher et al.* ABSTRACT, Lines 1-13).

Specifically, the registry has a variety of parts, including a namespaces table and an extensions table. All namespaces which participate in the user interface are registered in the namespace table. Using the namespace table of the registry, the user interface is able to associate the interface modules with the corresponding namespaces. Each namespace has a separate line in the namespace table associating that namespace with the corresponding interface module. The extensions table of the registry creates an association between namespaces. For instance, the foreign namespace can register itself to be associated with a target of the host namespace. Also an association can be created in the extension table between the target and the foreign namespace. (Col. 8, Lines 24-67, Col. 9, Lines –13 and 29-67).

In contrast to the inventions of Applicant's claims 1-24, however, the systems and methods taught by *Thatcher et al.* **do not teach a system or methods that contemplate the creation of cache having information relating to the classes of a class path, requesting a search on the class path, and searching the created cache to satisfy the search request.** Rather, *Thatcher et al.* teach a system and methods that aggregate namespaces independent of the

type of namespaces where each namespace does not require intimate knowledge of each other. In a particular implementation, *Thatcher* discloses the use of a registry that maintains a namespaces table an extensions table. The namespaces table maintains entries representative of namespaces, where the extensions table contains an association between the namespaces of the registry. (Col. 8, Line 24-67).

It is clear that the *Thatcher et al.* reference does not teach every limitation of the present invention as *Thatcher* fails to disclose the *creation of a cache having information relating to the classes of a class path, requesting a search on the class path, and searching the cache to satisfy the request.* As *Thatcher et al.* does not teach every limitation of the present invention, alone, or in combination with the other described references, a *prima facie* case of obviousness has not been made. Accordingly, Applicant respectfully submits that the present invention patentably defines over this cited reference alone or in view of the other herein described cited references. Further, Applicant respectfully requests the obviousness rejection be withdrawn on this basis.

Differences from *Ainsbury et al.*

Ainsbury et al. disclose an information platform that automates the collection of data, organizes the data into a library of information, and performs analysis using multiple-content types to provide participating users with relevant marketing information needed to make rapid and knowledgeable decisions. (Col. 2, Lines 41-45). The information platform is a client/server implementation that is subdivided into four major sections: Data Retrieval; Data Classification,

and Storage; Information Browsing, Query, Analysis, and Report Creation; and Integration. (Col. 6, Lines 37-67). In a preferred embodiment the sections contemplate the retrieval, classification, analysis of marketing data.

In contrast to the inventions of Applicant's claims 1-24, however, the systems and methods taught by *Ainsbury et al.* **do not teach a system or methods that contemplate the creation of cache having information relating to the classes of a class path, requesting a search on the class path, and searching the created cache to satisfy the search request.** Rather, *Ainsbury et al.* teach a system and methods to automatically retrieve data (e.g. marketing data), categorize the data, analyze, the data, and integrate the data for participating users. The *Ainsbury* reference teaches a search feature that may operate on the collected data to find desired elements. (Col. 22, Lines 36-52). However, the search feature described ***does not contemplate the creation of a cache having information relevant to classes of class path and performing a search on the cache to find desired classes as contemplated and claimed in the present application.***

Accordingly, Applicant respectfully submits that the present invention is not obvious over the *Ainsbury et al.* reference, alone, because the *Ainsbury et al.* reference does not teach all of the limitations of the present invention. Specifically, *Ainsbury et al.* do not teach systems and method that optimize the searching of class in a class path wherein a cache having information relevant to a class path is created, a search request for a particular class is generated, the search executed on the cache to find appropriate classes. Moreover, Applicant submits that the present invention is not obvious over the *Ainsbury* reference when combined with the other herein

described cited references since there is no motivation in the *Ainsbury* reference to combine with the other herein described references. *Ainsbury* is directed to an information platform for processing marketing information to provide to users to make informed decisions. The present invention, as stated, is directed to optimizing class identification for a class path. One of ordinary skill in the art would find it difficult to, refer to, yet alone, combine the teachings found in *Ainsbury* with the other herein described references to teach the present invention. Further, even if the *Ainsbury* reference is combined with the other herein described references, the present invention is not obvious over these herein described references since these herein described references, alone, or in combination **fail to teach every limitation of the present invention**. For these reasons, Applicant respectfully requests that the obviousness rejection be withdrawn.

Differences from *Timbol*

Timbol discloses a system to provide improved code generation capability that simplifies the task of application development. Specifically, a two-way code generation tool that may be used to not only allow automated generation of code, but also allow later automated generation of aspects of the code, even though the code itself has been edited. In a typical operation, the user may generate a Java-bean (i.e. a Java component) by invoking a wizard-based interface that implements methodology for automatically generating and managing a Java Bean or simply a “bean”. As part of the bean creation, a Java package manager may be employed to set certain attributes of the desired bean. (Col. 4, Lines 6-67, Col. 5, Lines 1-12).

In contrast to the inventions of Applicant’s claims 1-24, however, the systems and

methods taught by *Timbol* do **not** teach a system or methods that contemplate the creation of cache having information relating to the classes of a class path, requesting a search on the class path, and searching the created cache to satisfy the search request. Rather, *Timbol* teaches a system and methods to a Java development system that includes a client operating a virtual machine for executing programs. In particular, the client executes a compiled Java program which has been created by compiled a Java source code program or script with a Java compiler. At run time, a participating user may invoke an application wizard that assists the user in generating desired Java beans. The wizard, when invoked collects information about the files, class path, class loader location, and the like for the current project being developed. The underlying source file is parsed with a Java package manager to determine what classes are in the file, including the class corresponding to the Java bean being created. In contrast, the present invention contemplates the execution of search on multiple elements including a Java package manager to find desired classes of a class path. (Col. 21, Lines 10-67, and Col. 22, Lines 1-27).

Accordingly, Applicant respectfully submits that the present invention is not obvious over the *Timbol* reference, alone, because the *Timbol* reference does not teach all of the limitations of the present invention. Specifically, *Timbol* does not teach systems and method that optimize the searching of class in a class path wherein a cache having information relevant to a class path is created, a search request for a particular class is generated, the search executed on the cache to find appropriate classes. Moreover, Applicant submits that the present invention is not obvious over the *Timbol* reference when combined with the other herein described cited references since

there is no motivation in the *Timbol* reference to combine with the other herein described references. *Timbol* is directed to a system for developing computer programs that are created from components (e.g. Java beans), where a user is assisted by the system to create such components. The present invention, as stated, is directed to optimizing class identification for a class path. One of ordinary skill in the art would find it difficult to, refer to, yet alone, combine the teachings found in *Timbol* with the other herein described references to teach the present invention. Further, even if the *Timbol* reference is combined with the other herein described references, the present invention is not obvious over these herein described references since these herein described references, alone, or in combination **fail to teach every limitation of the present invention**. For these reasons, Applicant respectfully requests that the obviousness rejection be withdrawn.

Differences from *Fletcher*

Fletcher discloses a multiprocessing system having a hybrid cache control provided for each directory with each private CPU cache. The hybrid cache control selectively combines certain features found in a “store-in-buffer” cache and a “store through” cache to obtain a synergistic result of better system performance than when either type of cache is used alone. In operation flags are employed between the CPU and the cooperating caches to determine where and when to store the necessary bits in the appropriate caches. (Col. 2, Lines 39 – 56).

In contrast to the inventions of Applicant’s claims 1-24, however, the systems and methods taught by *Fletcher* **do not teach a system or methods that contemplate the creation**

of cache having information relating to the classes of a class path, requesting a search on the class path, and searching the created cache to satisfy the search request. Rather, Fletcher teaches a system and methods for a hybrid cache employed to optimize the storage of bits for a controlling CPU. In operation the hybrid cache provides a sharing flag with each line representation in each private cache director in a multiprocessor to uniquely indicate for each line in the associated cache whether it is to be handled like a “store-in-buffer” line or like a “store through” line. The present invention, in contrast, contemplates the use of a cache to contain information relevant to classes of a class path. The cache is then subsequently employed through a search feature to find desired classes. As such, the operations of the Fletcher cache and the cache of the present invention are very dissimilar.

Accordingly, Applicant respectfully submits that the present invention is not obvious over the Fletcher reference, alone, because the *Fletcher* reference does not teach all of the limitations of the present invention. Specifically, *Fletcher* does not teach systems and method that optimize the searching of class in a class path wherein a cache having information relevant to a class path is created, a search request for a particular class is generated, the search executed on the cache to find appropriate classes. Moreover, Applicant submits that the present invention is not obvious over the *Fletcher* reference when combined with the other herein described cited references since there is no motivation in the Fletcher reference to combine with the other herein described references. *Fletcher* is directed to a system and methods providing a hybrid cache control for a controlling CPU. The present invention, as stated, is directed to optimizing class identification

for a class path. One of ordinary skill in the art would find it difficult to, refer to, yet alone, combine the teachings found in *Fletcher* with the other herein described references to teach the present invention. Further, even if the *Fletcher* reference is combined with the other herein described references, the present invention is not obvious over these herein described references since these herein described references, alone, or in combination **fail to teach every limitation of the present invention**. For these reasons, Applicant respectfully requests that the obviousness rejection be withdrawn.

Differences from *Ferrel et al.*

Ferrel et al. disclose,

... a multimedia publishing system designed for creating publications that include components for design, authoring, distribution, viewing, search, personalization, and billing of on-line services. The major benefit of such a system is efficient distribution, content published separately from the layout, separation of responsibilities, hardware independence, automatically placed content and personalized titles.

Efficient distribution is achieved by separating the content and design which enables transmission of high-quality titles over low-speed communications links subject to loss of connectivity. Since the design of many titles remains reasonably static while the content changes on a regular basis, caching the layout designs minimizes the transmission of redundant data and optimizes the bandwidth use. Typically, content is transmitted quickly since it consists of tagged components, not the actual pages and controls themselves. (Col. 4, Lines 66-67, Col. 5 Lines 1-14).

In contrast to the inventions of Applicant's claims 1-24, however, the systems and methods taught by *Ferrel* **do not teach a system or methods that contemplate the creation of cache having information relating to the classes of a class path, requesting a search on the**

class path, and searching the created cache to satisfy the search request. Rather, Ferrel teaches a system and methods for a multimedia publishing system that employ caching object stores to improve on the delivery of media content. (Col 9, Lines 21-58). In the described embodiment *Ferrel* describes the ability multimedia system to provide objects from the cache so long as the cached objects are “not out of date or flushed.” (Col. 9, Lines 54-55). The present invention, in contrast, contemplates the execution of a date/time stamp check on the element having the cache information to ensure that the cache is up to date. (Claim 24 of the present application). The check for the present invention is an ***inclusive-type check*** wherein additional information is provided to update the cache. (Page 12, Lines 3-24). In contrast, the *Ferrel* invention, the date/time check is an ***exclusive-type*** check wherein the check is used to exclude objects that can be delivered from the cache. As such, the date/time check of the present invention operates differently and achieve different goals than the *Ferrel* date/time check. As such, *Ferrel* teaching is very dissimilar.

Accordingly, Applicant respectfully submits that the present invention is not obvious over the *Ferrel* reference, alone, because the *Ferrel* reference does not teach all of the limitations of the present invention. Specifically, *Ferrel* does not teach systems and method that optimize the searching of class in a class path wherein a cache having information relevant to a class path is created, a search request for a particular class is generated, the search executed on the cache to find appropriate classes. Moreover, Applicant submits that the present invention is not obvious over the *Ferrel* reference when combined with the other herein described cited references since

there is no motivation in the *Ferrel* reference to combine with the other herein described references. *Ferrel* is directed to a system and methods providing multimedia publishing where the format of the content can be separated and uploaded to a server by a publisher. The present invention, as stated, is directed to optimizing class identification for a class path. One of ordinary skill in the art would find it difficult to, refer to, yet alone, combine the teachings found in *Ferrel* with the other herein described references to teach the present invention. Further, even if the *Ferrel* reference is combined with the other herein described references, the present invention is not obvious over these herein described references since these herein described references, alone, or in combination **fail to teach every limitation of the present invention**. For these reasons, Applicant respectfully requests that the obviousness rejection be withdrawn.

Claim Analysis:

Claims 1 and 2:

Examiner has rejected claims 1 and 2 as being obvious over *Meyer et al.* and *Thatcher et al.* Examiner suggests that *Meyer et al.* discloses a method of generating a cache of information relating to the classes, requesting a search of the class path, and searching the cache to satisfy the requested search, and when combined with *Thatcher et al.*, that the Examiner suggests teaches the use of a class files database teach the present invention as claimed. However, as described above, *Meyer* and *Thatcher*, alone, or when combined do not teach every limitation found in claims one and two. ***Specifically, Meyer and Thatcher, alone, or in combination do not teach***

the creation of cache of having information pertinent to classes of a class path. Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claims 3 and 4:

Examiner has rejected claims 3 and 4 as being obvious over *Meyer* and *Thatcher* in view of *Ainsbury*. Examiner suggests that *Meyer et al.* discloses a method of generating a cache of information relating to the classes, requesting a search of the class path, and searching the cache to satisfy the requested search, and when combined with *Thatcher et al.*, that the Examiner suggests teaches the use of a class files database, and with *Ainsbury et al.* that Examiner suggests teaches the use of multiple elements, teach the present invention as claimed. However, as described above, *Meyer et al.*, *Thatcher et al.*, and *Ainsbury et al.*, alone, or when combined do not teach every limitation found in claims 3 and 4. ***Specifically, neither, Meyer, Thatcher, nor Ainsbury, alone, or in combination teach the creation of a cache having information pertinent to classes of a class path on which a search operates.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claims 5-6, 15-16, and 23:

Examiner has rejected claims 5-6, 15-16, and 23 as being obvious over *Meyer et al.* in

view of *Ainsbury et al.* Examiner suggests that *Meyer et al.* discloses the class path, wherein at least one of the elements has a cache of information sufficient to satisfy the request for that element, and when combined with *Ainsbury et al.*, that Examiner suggests teaches the use of multiple elements, teach the present invention as claimed. As described above, *Meyer* and *Ainsbury*, alone, or when combined do not teach every limitation found in claims 5-6, 15-16, and 23. ***Specifically, neither, Meyer nor Ainsbury, alone, or in combination teach generating and executing a search on a multi-element class path to find desired class elements.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claims 7 and 8:

Examiner has rejected claims 7 and 8 as being obvious over *Meyer* and *Thatcher* in view of *Ainsbury*. Examiner suggests that *Meyer et al.* in view of *Ainsbury et al.* discloses a method of locating classes in a multiple element class path, and when combined with *Thatcher et al.*, that the Examiner suggests teaches the use of ZIP files, teach the present invention as claimed. As described above, *Meyer et al.*, *Thatcher et al.*, and *Ainsbury et al.*, alone, or when combined do not teach every limitation found in claims 7 and 8. ***Specifically, neither, Meyer, Thatcher, nor Ainsbury, alone, or in combination teach generating and executing a search on a multi-element class path to find desired class elements.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be

withdrawn for these claims on this basis.

Claim 9:

Examiner has rejected claim 9 as being obvious over *Meyer* and *Ainsbury* in view of *Timbol*. Examiner suggests that *Meyer et al.* in view of *Ainsbury et al.* discloses a method of locating classes in a multiple element class path, and when combined with *Timbol*, that the Examiner suggests teaches the use of a Java Package Manager, teach the present invention as claimed. As described above, *Meyer et al.*, *Ainsbury et al.*, and *Timbol*, alone, or when combined do not teach every limitation found in claim 9. ***Specifically, neither, Meyer, Ainsbury, nor Timbol, alone, or in combination generating and executing a search on a multi-element class path to find desired class elements.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claims 10-13:

Examiner has rejected claims 10-13 as being obvious over *Meyer et al.* in view of *Fletcher*. Examiner suggests that *Meyer et al.* discloses the parsing of the class path, and when combined with *Fletcher*, that the Examiner suggests teaches a method of determining which elements are viable for caching and initiating creation of caches for those elements which are viable, teach the present invention as claimed. As described above, *Meyer* and *Fletcher*, alone,

or when combined do not teach every limitation found in claims 10-13. ***Specifically, Meyer and Fletcher, alone, or in combination do not teach the creation of cache for selected elements of a class path wherein certain class path elements are deemed viable.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claims 17-18 and 20-22:

Examiner has rejected claims 17-18 and 20-22 as being obvious over *Ainsbury et al.* in view of *Fletcher*. Examiner suggests that *Ainsbury et al.* discloses classes, multiple elements, and wrappers, and when combined with *Fletcher*, that the Examiner suggests teaches a viable cache, teach the present invention as claimed. As described above, *Ainsbury* and *Fletcher*, alone, or when combined do not teach every limitation found in claims 17-18 and 20-22. ***Specifically, Ainsbury and Fletcher, alone, or in combination do not teach the creation of cache for selected elements of a class path wherein certain class path elements are deemed viable.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claim 19:

Examiner has rejected claim 19 as being obvious over *Ainsbury et al.* and *Fletcher* in view of *Thatcher et al.* Examiner suggests that *Ainsbury et al.* in view of *Fletcher* disclose a means for parsing the multi-element class path, and when combined with *Thatcher et al.*, that the

Examiner suggests teach the use of ZIP files, teach the present invention as claimed. As described above, *Ainsbury*, *Thatcher*, and *Fletcher*, alone, or when combined do not teach every limitation found in claim 19. ***Specifically, Ainsbury Fletcher, and Thatcher, alone, or in combination do not teach the creation of cache for selected elements of a class path wherein certain class path elements are deemed viable and wherein the class elements comprise any of directories, ZIP files, and JACA package manager.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

Claim 24:

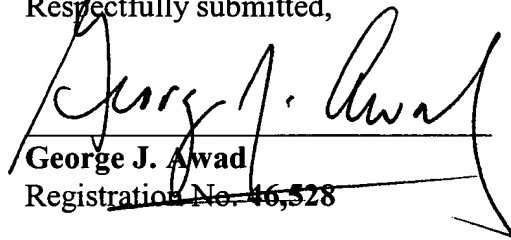
Examiner has rejected claim 24 as being obvious over *Meyer* and *Ainsbury* in view of *Ferrel*. Examiner suggests that *Meyer et al.* in view of *Ainsbury et al.* discloses a computer to perform a method of locating classes in a multi-element class path, and when combined with *Ferrel et al.*, that the Examiner suggests teach the use of a date/time stamp, teach the present invention as claimed. As described above, *Meyer et al.*, *Ainsbury et al.*, and *Ferrel*, alone, or when combined do not teach every limitation found in claim 24. ***Specifically, neither, Meyer, Ainsbury, nor Ferrel, alone, or in combination teach the execution of a time/date stamp on a class path having multi-elements wherein said elements contain a cache. The time/date stamp being employed to update the cache.*** Accordingly, a *prima facie* case of obviousness has not been made. Applicant respectfully requests that the obviousness rejection be withdrawn for these claims on this basis.

CONCLUSION

For all the foregoing reasons, Applicant respectfully submits that claims 1-24 patentably define over the prior art of record. Reconsideration of the present Office Action and an early Notice of Allowance are respectfully requested.

Attached hereto is a marked-up version of the changes made to the specification and claims by the current amendment. The attached page is captioned "Version With Markings To Show Changes Made."

Respectfully submitted,


George J. Awad
Registration No. ~~46,528~~

Date: December 5, 2001

WOODCOCK WASHBURN LLP
One Liberty Place - 46th Floor
Philadelphia, PA 19103
(215) 568-3100

VERSION WITH MARKINGS TO SHOW CHANGES MADE**In the Specification:**

Please replace the following paragraph found at Page 8, Lines 29-33; Page 9, Lines 1-14.

A class path is defined by use of an environment variable called `ClassPath`. `ClassPath` is used to define the order and places to search for classes when in an execution environment and in development environments. A class locator service, known as JPS, or the Java Package Service, exposes an API which describes the class path as a single class source, with methods such as `FindClass` and `EnumClasses` regardless of the current configuration of the class path. These methods find all classes with a given name in all or a specified package, and enumerate all classes of given name respectively. However, they currently search through all the classes identified by the class path. When a JPS instance is created, it is initialized with a list of class path elements. For each element, a root object is created which represents the contents of the element. This object exposes an interface called `IRoot`, which has simple class location/enumeration methods, such as `DoesClassExist`, `GetClass`, and `EnumClasses`, each of which do what their names imply. Depending on the type of the element, a different implementation of a root object is created. For directory elements, a direct root object is created, whose `IRoot` method implementations result in direct searches of the [rot's] root's underlying file system directory. For other elements, a root implementation that performs caching is created.